Design-by-Contract for *Flexible* Multiparty Session Protocols

Lorenzo Gheri @ Imperial College
Ivan Lanese @ Focus Team, University of Bologna/INRIA
Neil Sayers @ Imperial College & Coveo Solutions Inc.
Emilio Tuosto @ GSSI
Nobuko Yoshida @ Imperial College

ECOOP Berlin 2022

Take-home message

Take-home message

Choreography Automata

A model of choreographies of message-passing systems featuring

- selective participation
- deadlock and lock freedom by construction
- design-by-contract: constrain payloads of communications

Take-home message

Choreography Automata

A model of choreographies of message-passing systems featuring

- selective participation
- deadlock and lock freedom by construction
- design-by-contract: constrain payloads of communications

CAScr (`https://github.com/Tooni/CAScript-Artifact`)

A tool chain for

- top-down choreographic development
- validating protocols via choreography automata
- TypeScript web programming via API generation

Take-home message

Choreography Automata

A model of choreographies of message-passing systems featuring

- selective participation
- deadlock and lock freedom by construction
- design-by-contract: constrain payloads of communications

CAScr (`https://github.com/Tooni/CAScript-Artifact`)

A tool chain for

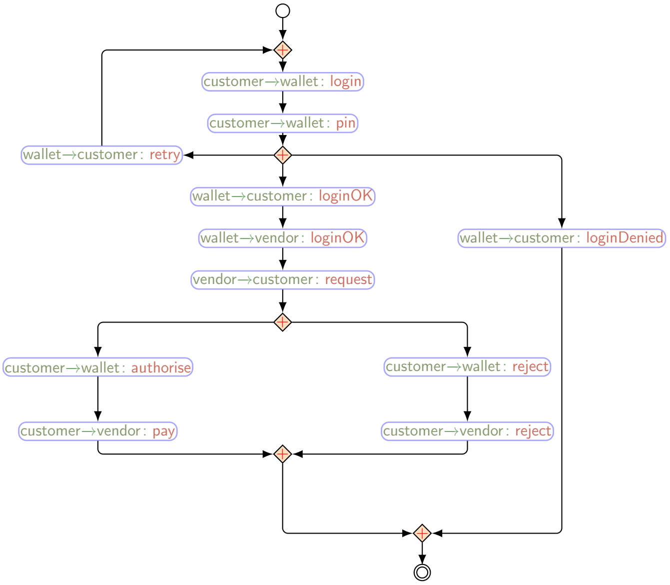
- top-down choreographic development
- validating protocols via choreography automata
- TypeScript web programming via API generation

Check out our paper or get in touch for details...

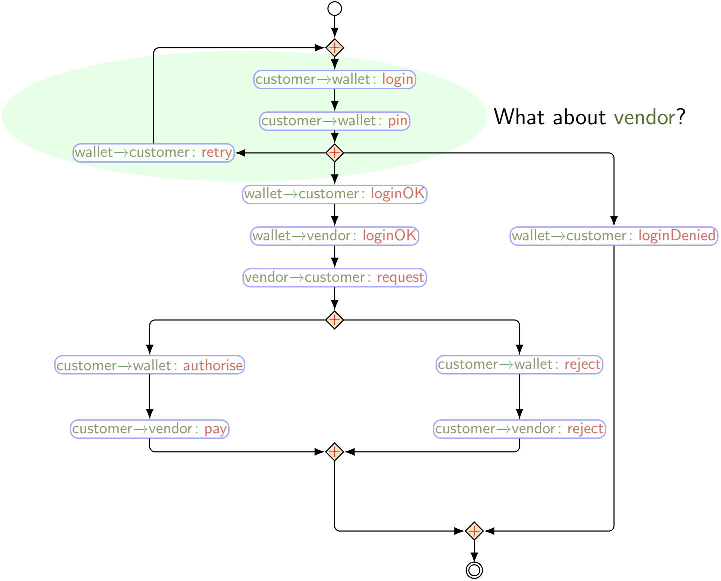
– Prologue –

[Choreographies, informally]

The online-wallet protocol

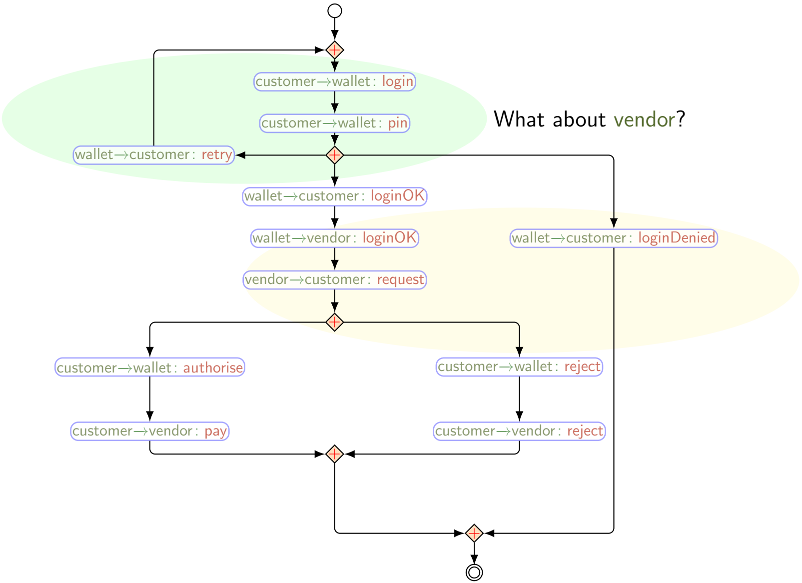


The online-wallet protocol ...some modelling problems



What about vendor?

The online-wallet protocol ...some modelling problems



What about vendor?

What about payloads?

Top-down model-driven development

Choreography = Global spec + Local spec

Quoting W3C:

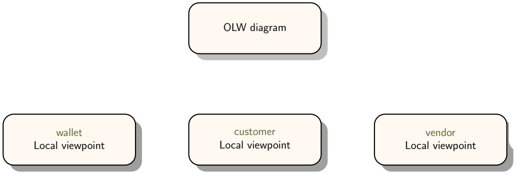
"[...] a contract [...] of the common ordering conditions and constraints under which messages are exchanged [...] from a global viewpoint [...]
Each party can then use the global definition to build and test solutions [...]
global specification is in turn realised by combination of the resulting local systems"

Top-down model-driven development

Choreography = Global spec + Local spec

Quoting W3C:

*"[...] a contract [...] of the common ordering conditions and constraints under which messages are exchanged [...] from a global viewpoint [...]
Each party can then use the global definition to build and test solutions [...]
global specification is in turn realised by combination of the resulting local systems"*



well-formedness

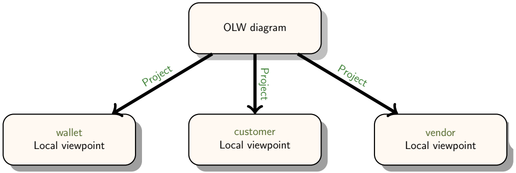
specs,not code

Top-down model-driven development

Choreography = Global spec + Local spec

Quoting W3C:

"[...] a contract [...] of the common ordering conditions and constraints under which messages are exchanged [...] from a global viewpoint [...]
Each party can then use the global definition to build and test solutions [...]
global specification is in turn realised by combination of the resulting local systems"



well-formedness

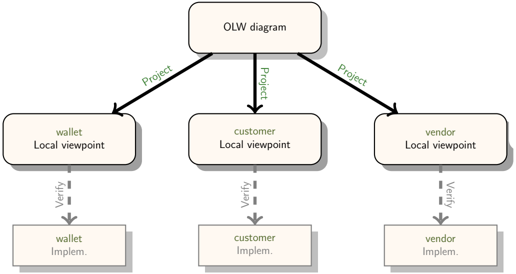
specs,not code

Top-down model-driven development

Choreography = Global spec + Local spec

Quoting W3C:

*"[...] a contract [...] of the common ordering conditions and constraints under which messages are exchanged [...] from a global viewpoint [...]
Each party can then use the global definition to build and test solutions [...]
global specification is in turn realised by combination of the resulting local systems"*



well-formedness

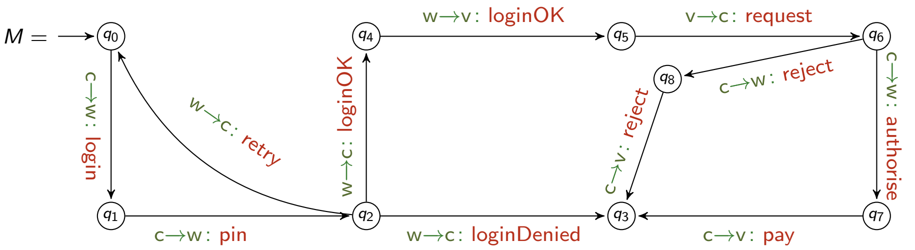
specs,not code

– Act I –

[Choreography Automata]

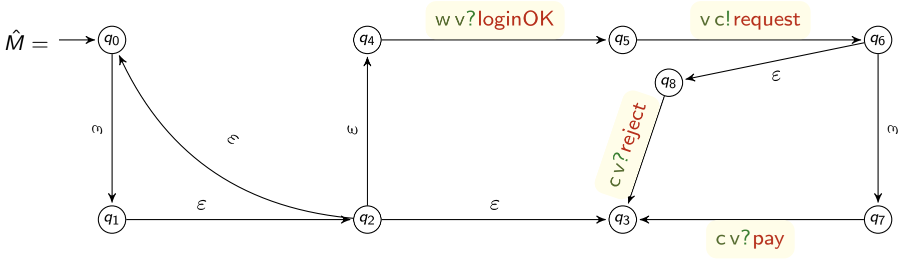
Our global & local specs

Choreography automata: Interaction, globally

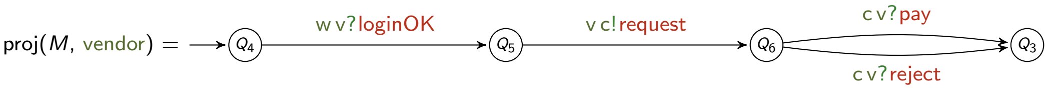


Our global & local specs

Intermediate automata: from interactions to communications

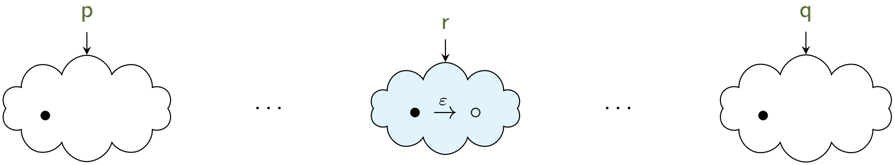


Communicating finite-state machines: Communication, locally



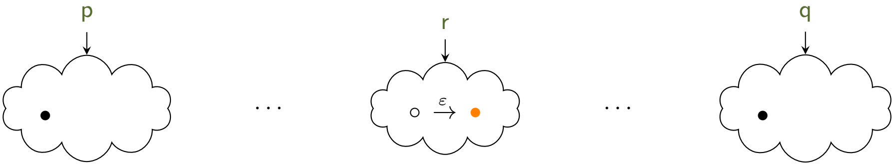
Semantics of CFSMs

Internal step: $S \xrightarrow{\varepsilon} S'$



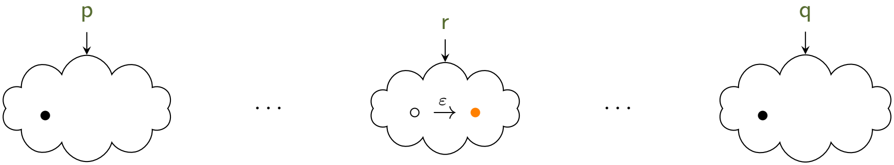
Semantics of CFSMs

Internal step: $S \xrightarrow{\varepsilon} S'$

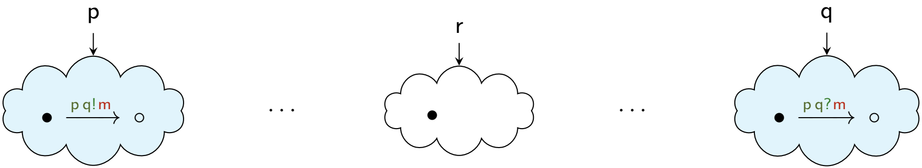


Semantics of CFSMs

Internal step: $S \xrightarrow{\varepsilon} S'$

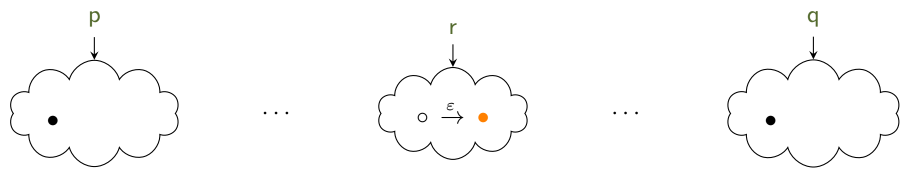


Interaction: $S \xrightarrow{\mathsf{p} \rightarrow \mathsf{q} : \mathsf{m}} S'$

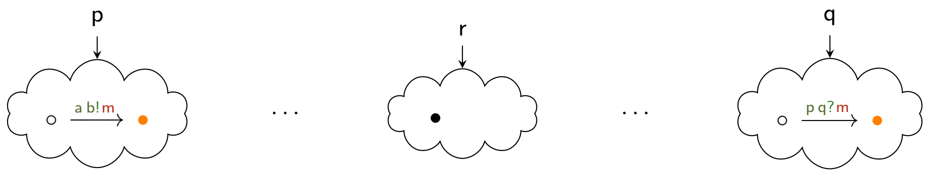


Semantics of CFSMs

Internal step: $S \xrightarrow{\varepsilon} S'$



Interaction: $S \xrightarrow{p \rightarrow q : m} S'$



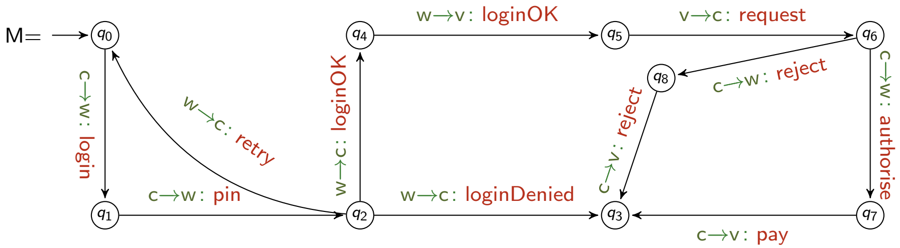
Projections preserve semantics

Theorem. Choreography automata are bisimilar to their projections

$\Longrightarrow$ traces equivalence

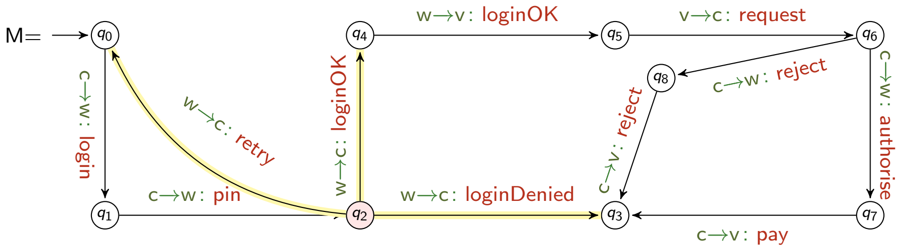
Flexibility by example

Selective participation in OLW



Flexibility by example

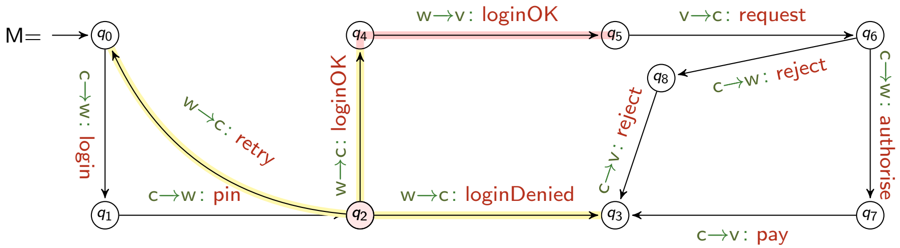
Selective participation in OLW



- at q_2 wallet and customer aware from the very beginning

Flexibility by example

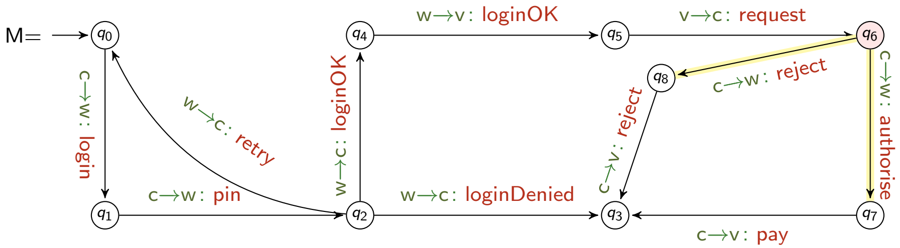
Selective participation in OLW



- at q_2 wallet and customer aware from the very beginning
  - vendor involved on one branch only, but that's fine: wallet is aware

Flexibility by example

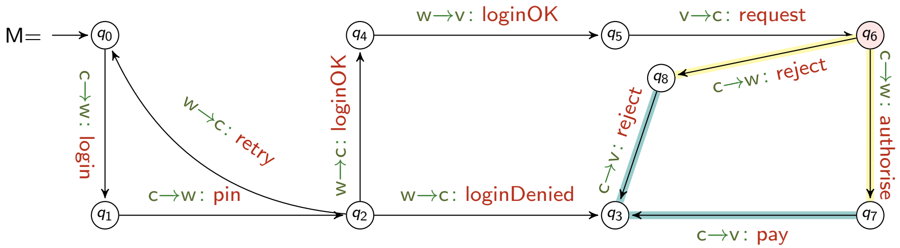
Selective participation in OLW



- at q_2 wallet and customer aware from the very beginning
  - vendor involved on one branch only, but that's fine: wallet is aware
- at q_6 wallet and customer aware from the very beginning

Flexibility by example

Selective participation in OLW



- at q_2 wallet and customer aware from the very beginning
  - vendor involved on one branch only, but that's fine: wallet is aware
- at q_6 wallet and customer aware from the very beginning
  - vendor eventually informed by customer on each branch

Theorems

Correctness by construction

Theorem. Projections of well-formed choreography automata are deadlock-free

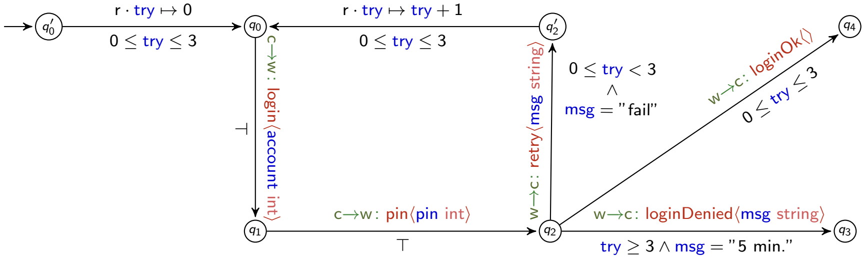
Theorem. Projections of well-formed choreography automata are lock-free

– Act II –

[Asserted Choreography Automata]

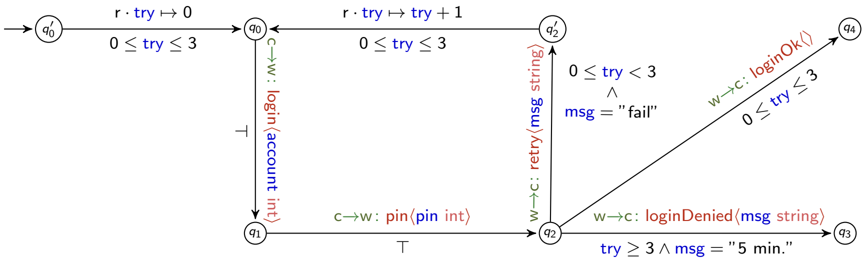
DbC vs. choreography automata

Asserting (an excerpt of) OLW



DbC vs. choreography automata

Asserting (an excerpt of) OLW



Consistency

- history senesitiveness: in $q \xrightarrow[A]{\lambda} q'$, A predicates on *known* variables
- temporal satisfiability: the conjunction of the predicates on a path is satisfiable
- well-formedness of the underlying choreography automaton

Theorems

Projections are a bit more complicated than for choreography automata

On consistent asserted choreography automata

Theorem. Asserted choreography automata are weakly bisimilar to their projections

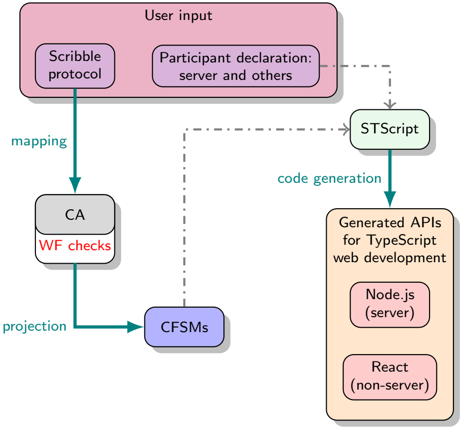
$\Longrightarrow$ trace equivalence

Theorem. Projections of well-formed asserted choreography automata are deadlock-free

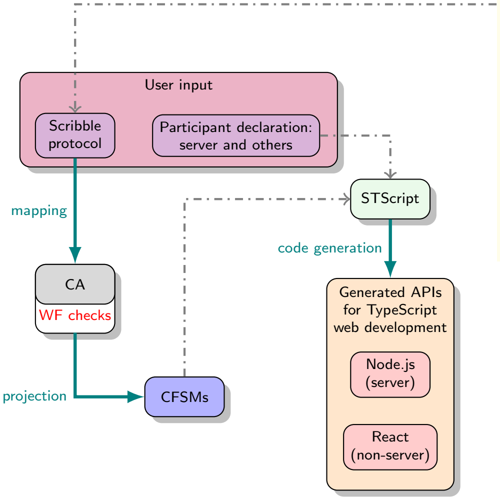
– Act III –

[CAScr]

Architecture of CAScr

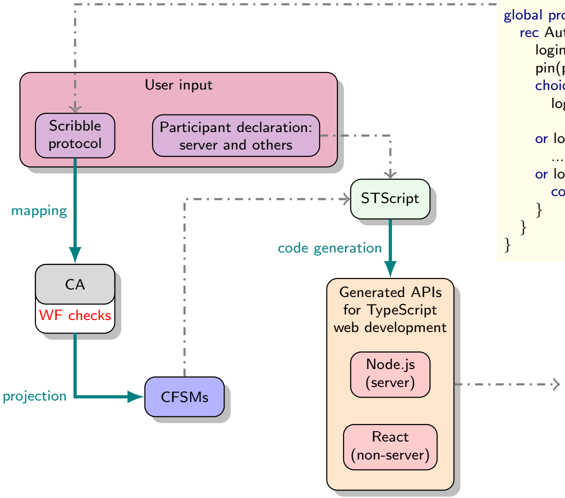


Architecture of CAScr

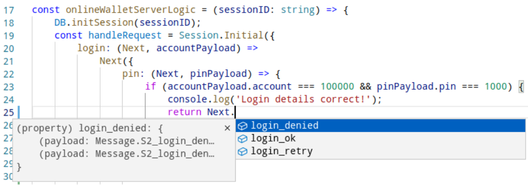



```
global protocol OnlineWallet(role wallet, role customer, role vendor) {
  rec AuthLoop {
    login(account: int) from customer to wallet;
    pin(pin: int) from customer to wallet;
    choice at wallet {
      login_ok() from wallet to customer;
        ...
    or login_denied(msg: string) from wallet to customer;
      ...
    or login_retry(msg: string) from wallet to customer;
      continue AuthLoop;
    }
  }
}
```

Architecture of CAScr




```
global protocol OnlineWallet(role wallet, role customer, role vendor) {
  rec AuthLoop {
    login(account: int) from customer to wallet;
    pin(pin: int) from customer to wallet;
    choice at wallet {
      login_ok() from wallet to customer;
        ...
    or login_denied(msg: string) from wallet to customer;
      ...
    or login_retry(msg: string) from wallet to customer;
      continue AuthLoop;
    }
  }
}
```



Multiparty global types

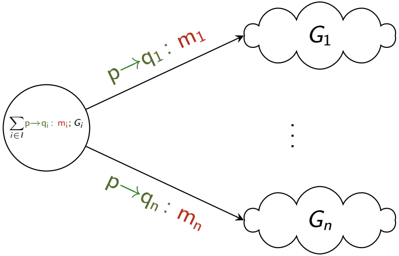
$\boxed{\text{Syntax}}$

$$G \quad ::= \quad \sum_{i \in I} \mathsf{p} \to \mathsf{q}_\mathsf{i} : \mathsf{m}_\mathsf{i}; G_i \qquad \mu \mathsf{r}.G \qquad \mathsf{r} \qquad \mathtt{end}$$

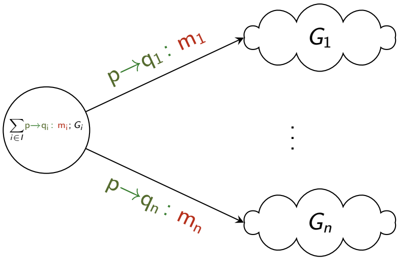
$\boxed{\text{Semantics}}$

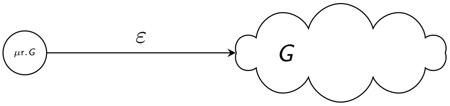
$$\sum_{i \in I} \mathsf{p} \to \mathsf{q}_\mathsf{i} : \mathsf{m}_\mathsf{i}; G_i \xrightarrow{\mathsf{p} \to \mathsf{q}_\mathsf{j} : \mathsf{m}_\mathsf{j}} G_j \ (j \in I) \qquad\qquad \frac{G[\mu \mathsf{r}.G/\mathsf{r}] \xrightarrow{\alpha} G'}{\mu \mathsf{r}.G \xrightarrow{\alpha} G'}$$

From global types to choreography automata

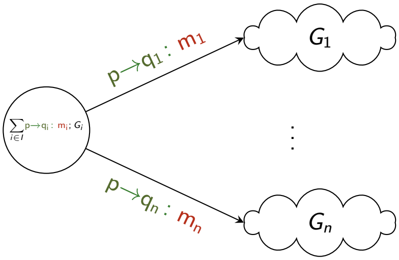


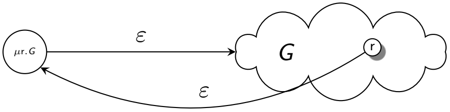
From global types to choreography automata



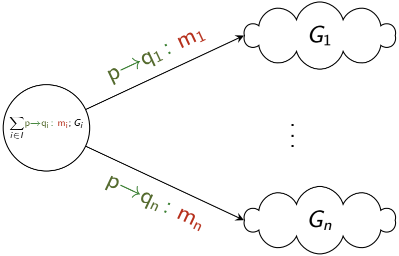


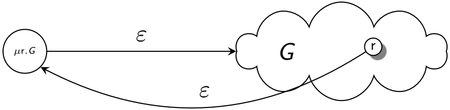
From global types to choreography automata





From global types to choreography automata





CAScr

- computes the mapping above
- checks well-formedness of the resulting choreography automaton
- generates the TypeScript API of each participant

– Epilogue –

[...]

Summing up

Choreography Automata (with assertions)

A theory of choreographies

- with increased expressiveness
- supporting DbC
- providing a basis for (enhanced) tool support for TypeScript web programming

Plans

- Consider asynchronous communications
- Applications:
  - inferring a (local) models from APIs and
  - checking their conformance against projections of a global spec

[Thank you!]